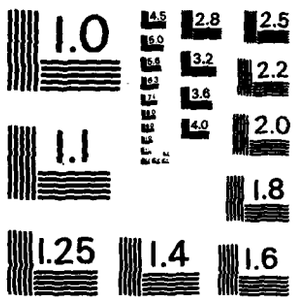
AD A137 068 PARALLEL LOGIC PROGRAMMING AND ZMOB AND PARALLEL 1/1
SYSTEMS SOFTWARE AND HAR..(U) MARYLAND UNIV COLLEGE
PARK DEPT OF COMPUTER SCIENCE J MINKER ET AL. SEP 83
UNCLASSIFIED AFOSR-TR-84-0004 AFOSR-82-0303 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFOSR-TR- 84-0004	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): PARALLEL LOGIC PROGRAMMING AND ZMOB AND PARALLEL SYSTEMS SOFTWARE AND HARDWARE	5. TYPE OF REPORT & PERIOD COVERED: INTERIM, 1 JUN 82-15 SEP 83
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Jack Minker and Mark Weiser	8. CONTRACT OR GRANT NUMBER(s): AFOSR-82-0303
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Computer Science, University of Maryland, College Park MD 20742	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS: Mathematical & Information Sciences Directorate, Air Force Office of Scientific Research /NM, Bolling AFB DC 20332	12. REPORT DATE: SEP 83 13. NUMBER OF PAGES: 10
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
During this period the investigators had four papers accepted for publication in refereed conference proceedings, two papers appeared in an invited workshop and one technical report and one technical note were written. Titles include, "Interfacing predicate logic languages and relational databases," "A parallel inference system based on logic," "A theory of forward and backward tracking," and "Design of the intensional database system of the ZMOB parallel problem solver." The hardware side of the project is progressing but ZMOB does not exist yet as a working system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AFOSR-TR- 84-0004

(3)

Parallel Logic Programming and ZMOB

and

Parallel Systems Software and Hardware

A Summary of Work Performed for the Air Force Office

of Scientific Research

June 1982-September 15, 1983

by

Professor Jack Minker

and

Asst. Professor Mark Weiser

Department of Computer Science

University of Maryland

College Park, Maryland 20742

84 01 19 140

1. Introduction

The purpose of this letter is to discuss our research into parallel systems software and hardware, and parallel problem solving, initiated under Air Force Grant AFOSR-82-0303. Under the current grant, a detailed design and partial implementation of a parallel problem solving system, PRISM (*parallel inference system*), based on logic was achieved. The PRISM requires that the ZMOB parallel processor be available for use. In addition, systems software and hardware have been developed. It is estimated that ZMOB will become available for use during the Fall of 1983. Hence, a full test and debugging of PRISM cannot be achieved under the current grant.

At the end of the current grant we expect to have accomplished, as a minimum, all of the objectives proposed. That is, in the area of parallel problem solving, the initial PRISM has been fully designed; individual programs have been implemented and tested in a non-parallel environment; and investigations have been made into extensions to the initial design. As a consequence of the work, six papers were accepted for publication in conference proceedings, and one technical note was written. Other papers are in progress and will be completed before the end of the current grant.

2. Accomplishments on Efforts During Period June 30,1982-September 15,1983

In section 2.1 we discuss work accomplished in parallel problem solving, and in section 2.2 we discuss work in parallel systems software and hardware. We describe the accomplishments made as of August 15 and those expected by the end of the grant,September 15, 1983.

2.1 *ZMOB and Parallel Problem Solving*

During the period June 30,1982-September 15,1983, we proposed that three major tasks be accomplished:

(a) Design of an Initial Parallel Logic Problem Solver.

(b) Implementation of the initial design.

(c) Investigate extensions to the initial design.

In the following three sections we describe progress made on these efforts. In the fourth section, we describe publications and reports written, that describe the work that has been completed.

2.1.1 *Design of an Initial Parallel Logic Problem Solver*

The functional specification of a system termed PRISM (*P*arallel *I*nference *S*yste*m*) was essentially completed before the start of the effort. A document termed "Functional Specifications of the ZMOB Problem Solver" was written and completed during the period of the grant and provides the complete functional specification of the system.

The detailed design of the functional specification is completed. Some modifications may be made as the system undergoes implementation.

2.1.2 Implementation

The implementation of PRISM requires the availability of ZMOB. The ZMOB system is not yet fully available to support the complete implementation of PRISM. A stand alone Z80A system that was submitted for bids during May 1982 has been approved by the University for purchase and delivery was made in February 1983.

Substantial progress has been made on implementation in spite of the above equipment status. The status of the major parts of the design is as follows.

(1) *User interface with VAX*

The program provides the interface between the user and VAX, the host computer for PRISM. The program has been implemented and is tested. A full test will require ZMOB.

(2) *Extensional Database (EDB) for Moblets* The Extensional Database (EDB) resides on moblets. An assembly language program has been implemented and runs on a single Z80A. When the entire relational table resides on a single moblet, the EDB code responds to queries. The program is partially tested and the code that supports the distribution of the EDB on several machines exists. Testing requires the availability of ZMOB.

(3) *Extensional Database for VAX*

The EDB is constructed by the user in the VAX machine. The database supplied by the user is stored initially in the VAX. Syntax checks are performed on the input data, and errors are reported to the user. The data is then indexed, and formatted for transmission to the appropriate number of moblets required. Implementation of this portion of the system is in progress and is expected to be

substantially completed prior to September 15, 1983.

(4) *Intensional Database (IDB) for ZMOB and Monitor*

The Intensional Database (IDB) stores the procedures in the system. The initial design does not permit the number of procedures to exceed the size of a moblet. The IDB has been implemented in Pascal and is being tested on the VAX. The program has been converted to C for the moblets. Debugging of the code will require ZMOB. The IDB monitor has been implemented and is being tested.

(5) *Intensional Database (IDB) for VAX*

As with the EDB, the IDB is entered into the VAX by the user. The program does syntax checks on the input data, coordinates with EDB data provided by the user, indexes the procedures and prepares the programs and data for transmission to the moblets. The program has been implemented and tested. It has been interfaced with the user interface program. It must be tested in conjunction with the ZMOB.

(6) *Problem Solving Machine (PSM)*

The Problem Solving Machine is the central portion of PRISM. A stripped-down version of the design has been implemented in PROLOG and is running on the VAX. The full PSM program has been implemented in PROLOG and converted to C, so as to improve efficiency. The PROLOG program will then serve as a high level operational and axiomatic definition of the PSM. The code has to be debugged and requires the availability of ZMOB.

(7) *Communication Primitives*

The communication primitives have been implemented and require ZMOB for a full test.

To further the implementation, a portion of ZMOB is simulated on the VAX and interfaces with the existing programs. A primitive system consisting of the user-VAX interface, the IDB programs, and the stripped-down version of the PSM, on a simulated ZMOB belt, has been tested. This has permitted us to test several portions of the system together rather than in isolation.

2.1.3 Investigate Extensions to the Initial Design

The primary emphasis since the award of the current grant has been directed towards the design and implementation of PRISM. Simultaneously extensive investigations were made on a number of topics as described below. We expect to have some of these efforts in an advanced stage of completion by September 1983 and others will require substantial work during the next year's proposed effort.

(a) *Extensions to the Problem Solving Machine*

Considerable progress has been made on this subject. Our initial design has been extended to include a limited amount of AND-parallelism. Further work will be performed on achieving full AND-parallelism.

Consideration is being given to the control structure in general. A theory of backtracking has been developed for function-free problems. A technical report has been written that describes how one can take advantage of integrity constraints.

Work on control structures will continue into the next year's work.

(b) *IDB Extensions*

Work is starting on the design for the IDB which will permit extension to procedures that exceed the size of a single moblet.

(c) *Backend Database Machines and Peripheral Devices*

A study has started on backend database machines to determine their availability. Once this is done, considerations will be given to interfacing with such systems. Considerations as to how to interface with a relational database machine have been given as described in Section 2.1.4, below.

(d) *User and System Considerations*

To be an effective tool in a research environment, additional features must be added to PRISM. Preliminary considerations are being given to the gathering of statistics; providing the user the ability to view the proof tree structure as it is evolving; and recovery in the event of system failures. We expect to have a preliminary functional specification of these areas by September 1983.

2.1.4 Publications and Reports

As a consequence of work performed on this effort, four papers were accepted for publication in refereed conference proceedings, two papers appeared in an invited workshop, one additional technical

report was written and one technical note.

The publications and reports cover a number of different topics : functional description and detailed design; PRISM system and alternative aspects; control structures; and interfacing logic programs and databases.

(a) *Functional Description and Design*

In [1] a functional description is provided of the PRISM. The description covers the EDB, the IDB, the IDB monitor, the user interface and the PSM. A detailed description of the design of the IDB is provided in [2]. These reports are intended for internal use and will not be submitted to journals.

(b) *PRISM System*

Aspects of the PRISM appear in a number of papers and reports. In [4] we present the initial ideas that led to the ultimate design of the system. The paper [5] is a condensation and minor revision of the earlier report. Aspects of the PRISM system are covered in papers [2], [6], and [8]. The heart of the PRISM, the problem solving machine (PSM) is described in [6].

(c) *Control Structures*

A theory of intelligent forward and backward tracking is provided in [7]. The theory applies to function-free problems and describes how lemmas and integrity constraints may be used during the search process.

(d) *Interfacing Logic Programs and Databases*

Various possible ways in which a database may be interfaced with logic programs are discussed in [3]. Among the options described is the combining of object language and meta-language statements.

2.2 Work Completed

The Zmob computer is composed of 256 Z80 computers connected on an intelligent high-speed ring bus called the "conveyer belt". It has been described extensively elsewhere. Each Z80 has 64k bytes of memory, a parallel port, a serial port, and a floating point chip in addition to its conveyer belt connection. Initially Zmob will be completely dependent on a host computer for downloading of programs and for mass storage. The Host will be the Department of Computer Science's Vax-11/780, or a dedicated

Vax purchased under grants pending under the DoD special equipment program and the National Science Foundation.

Individual Z80's with associated hardware are called "moblets". A moblet consists of a Z80 and a "mailstop" or "mailbox". Mailboxes are tied together on the "belt". The mailstops associated with host communications are "Vaxstops".

2.2.1 Hardware

The Zmob basic design has been checked in two-moblet versions both wire-wrapped and on printed circuit boards. Multiple processors can run independently, as many as necessary (up to 16 have been tested). A serious glitch was found in the processor card in January 1983, and since then there have been no processor failures.

Hardware debugging has focused on communications between the processors. The four components of the communication system--the master clock, the clock cables and backplane, the backplane interconnect cables, and the mailstop boards--have each undergone several revisions. The clock circuit is being tuned to provide optimal timing between the clock and index pulses to the moblets. Several versions of clock cable are being tried, including coax, twisted pair, and shielded twisted pair. The clock cables are critical because they must run for several yards to distribute clock to all the processors. Deglitching capacitors on the backplane have eliminated some noise, and pull-up resistors on the mailstop boards have been adjusted for better noise tolerance. A logic analyzer has been rented to better view multiple mailstop signals.

Communication is reliable in one direction along a single backplane. A complete communications loop can be achieved for several seconds without a failure. The problems are not unexpected and primarily involve noise and timing relations with the master clock.

2.2.2 Software

There are three main projects looking with applications for Zmob. These are the Computer Vision project with Professors Rosenfeld and Davis, the Distributed Numerical Computation project with Professors Stewart and O'Leary, and the Distributed Problem Solving project with Professor Minker. Each plans to use Zmob to explore parallel processing algorithms in their specific domains, and each faces the

problem of operating system support for coordinating the processes and accessing the powerful Zmob hardware

These three projects need similar basic operating system utilities. The projects' implementation strategies have been driven by the Zmob conveyer belt hardware design but none of the projects envisions working at the machine code level at which the conveyer belt can be directly accessed. Each requires high-level language access to the full conveyer belt capability.

The system utilities for use by all three projects are under development. In particular, the following software has been completed and is fully documented on-line the Computer Science Department's Vax computer:

1. c - Run Whitesmiths C compiler for z80/zmob
2. belt - primitive routines for zmob mail stops
3. lo1 - IO library for zmob.
4. zfasl - Load a zmob program at 9600 baud
5. zgo - transfer a program to the zmob, and go
6. zload - load the zmob's memory

Software working but for which documentation is still being readied includes:

(1) Z80 Simulator This emulates the basic Z80 processor without the Zmob environment. It was the first emulator to be completed and provided initial experience with the Z80 environment. It is a sufficiently detailed emulation to permit running CP/M and Prolog, a capability much used by the distributed problem-solving group.

(2) Communications Simulator. This emulates multiprocessing and use of the mailstop and conveyer belt environment but executes Vax C code rather than Z80 machine language. It performs multiprocessing using basic Unix capabilities. The interface to the conveyer belt uses the same high level language calls that will be available on the moblets. Check out has been via an implementation of a solution to the dining philosophers problem.

(3) Communications Activity Display. This system accepts a stream of input describing the belt activity and displays on a color graphics display the resultant belt status. The input stream is now generated by the communications simulator (see above) and will eventually be generated by the 'monitor mail stop' on the actual Zmob (see below).

(4) System Debugging Harness. This permits debugging new low level system software for inclusion in Zmob. It allows the tester to manually create conditions which in actual practice would be quite rare, thus testing the robustness of the system. It consists of three parts: (a) the system software under test, (b) the moblet simulation system, and (c) the user interface display connected to both (a) and (b). The system software under test believes that it is running within a moblet, as emulated by the moblet simulation system. In fact, however, all attempts at communication are simply displayed to the user via (c). The user can then specify any arbitrary response on the part of the hardware, including timing relationships. The right half of the user's display is reserved for interaction about attempts at moblet communication. The left half of the display can be used by the system under test to display internal status of interest to the user.

(5) Multiple-window Communications Interface. This is an adaptation of the maryland window shell to problems of communications with the Zmob. The window shell allows arbitrarily sized and positioned rectangular windows on a single CRT screen, each functionally equivalent to an independent terminal. A typical application is to open several windows and run a communication process to a different moblet in each window.

8. References

[1] Minker, J., et al., "Functional Description of the ZMOB Parallel Problem Solving System", Technical Note 1, Department of Computer Science, University of Maryland, December 1982.

[2] Cao, D., "Design of the Intensional Database System of the ZMOB Parallel Problem Solver", Technical Report Computer Science Department, University of Maryland, 1982.

[3] Chakravarthy, U.S., Minker, J. and Tran, D., "Interfacing Predicate Logic Languages and Relatic Databases". *Proceedings of the First International Logic Programming Conference*, September 17, 1982, Faculte des Sciences de Luminy Marseille, France, 91-98.

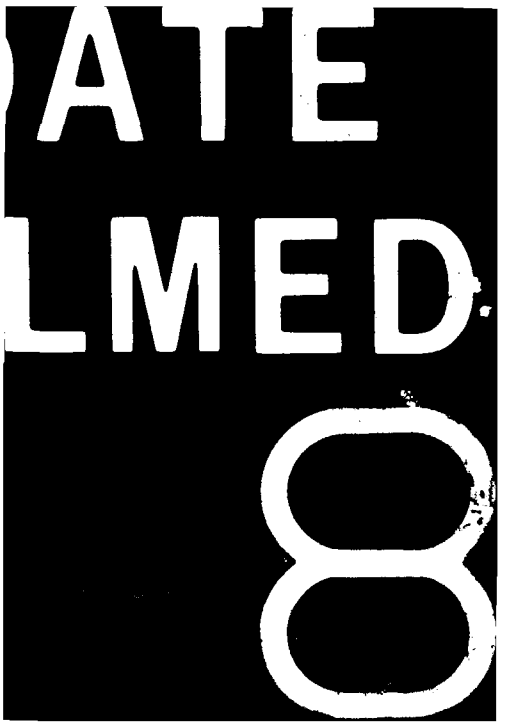
[4] Eisinger, N., Kasif, S., and Minker, J., "Logic Programming : A Parallel Approach". Technial Report TR-1124, Computer Science Department, University of Maryland, December 1981.

[5] Eisinger, N., Kasif, S., and Minker, J., "Logic Programming : A Parallel Approach", *Proceedings of the First International Logic Programming Conference*, September 14-17, 1982, September 14-17,1982, Faculte des Sciences de Luminy Marseille, France, 71-77.

[6] Kasif, S., Kohli, M., and Minker, J., "PRISM - A Parallel Inference System Based on Logic", Technical Report TR-1243, Computer Science Department, University of Maryland, February 1983.

[7] Kohli, M., and Minker, J., "A Theory of Intelligent Forward and Backward Tracking", Technical Report (in preparation), Computer Science Department, University of Maryland, March 1983.

[8] Minker, J., et al.,"Parallel Problem Solving on ZMOB", Proceedings of Trends and Applications 83, Washington, D.C.,1983.


ATE
LMED
8